I/O Performance

- What is performance?
  - With CPU: Iron Law
  - With I/O
    - Applications have different I/O needs
- Supercomputers read and write 1G of data
  - High data throughput
- Transactions processing does many independent, small I/O ops.
  - Fast I/O throughput (# of I/Os per sec)
- File systems
  - Fast response time, locality

Motivation

- I/O needed for
  - To/from users
  - To/from computers
  - To/from non-volatile storage media
- I/O Performance matters
- Total time = CPU + I/O - overlap
  - 10 + 4 - 4 = 10; 1x
  - 5 + 4 - [0,4] = [9,5]; [1.1x, 2x]
  - 1 + 4 - [0,1] = [5,4]; [2x, 2.5x]

I/O Device Characteristics

<table>
<thead>
<tr>
<th>Device</th>
<th>I or O?</th>
<th>Partner</th>
<th>Data Rate KB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse</td>
<td>I</td>
<td>Human</td>
<td>0.01</td>
</tr>
<tr>
<td>Graphics Display</td>
<td>O</td>
<td>Human</td>
<td>60,000</td>
</tr>
<tr>
<td>Modem</td>
<td>I/O</td>
<td>Machine</td>
<td>2-8</td>
</tr>
<tr>
<td>LAN</td>
<td>I/O</td>
<td>Machine</td>
<td>500-6000</td>
</tr>
<tr>
<td>Tape</td>
<td>Storage</td>
<td>Machine</td>
<td>2000</td>
</tr>
<tr>
<td>Disk</td>
<td>Storage</td>
<td>Machine</td>
<td>2000-10000</td>
</tr>
</tbody>
</table>
Disk Trends

- Disk Trends
  - \$/MB decreasing
    - 2004: $110 / 80GB (IDE/ internal retail)
    - Less than 1c/MB (was 10c/MB in 1997)
  - OEM prices lower
  - Disk diameter 14" -> 1.8" -> 1"
  - Seek time down ~10ms
  - Rotation speed unchanged (~7200 rpm)
  - Xfer rates up

Magnetic Disks

- Stack of platters
  - Two surfaces/platter
- Concentric tracks
  - Several sectors/track
- Disk Access Time = Queue time + service time

Recap: Disk Access Time

- Disks R'us™ has a GenX disk with the following parameters
  - Average seek time = 10ms
  - Rotation speed = 5400 rpm
  - Xfer rate = 16MB/s
  - Disk-block size = 4KB
  - Controller overhead = 3ms
- They introduce a GenY disk technology with rotation speed of 7200 rpm which they advertise as being 50% faster
- If average queue time is 0ms, what is the true speedup of GenY over GenX when reading a randomly chosen disk-block.
**Disk Access Time**

- GenX access time:
  - Queue time + controller overhead + seek time + rotational latency for ½ rotation + Xfer time
  - \[0 + 3 \times 10 + (0.5 \times 60/5400 \times 10^3) \text{ms} + (4K/16M \times 10^3) \text{ms}\]
  - \[0 + 3 \times 10 + 5.6 + 0.25\]
  - \[18.85 \text{ms}\]

- GenY access time:
  - \[0 + 3 \times 10 + (0.5 \times 60/7200 \times 10^3) \text{ms} + (4K/16M \times 10^3) \text{ms}\]
  - \[0 + 3 \times 10 + 4.1 + 0.25\]
  - \[17.35 \text{ms}\]

- Speedup: \[18.85/17.35 = 1.086 = 8.6\%\]

---

**Redundant Array of Inexpensive Disks**

- What if we want to store 100 disks
- MTTF: 5 yr/100 = 18 days
  - RAID 1 = mirror = full data redundancy = 100% overhead
  - RAID 3 = bit wise parity = small overhead
    - Dedicated parity disk
  - RAID 5 = block wise parity = small overhead + small writes
- Google for RAID details

---

**Connecting I/O to the CPU**

- Many I/O devices with vastly different datarates
- I/O device economics
  - Need standards
    - Large number of peripheral device producers
    - Multiple business entities involved favors longer lasting standards
- Let’s examine several connection strategies

---

**Advantages of Buses**

- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways
  - Manage complexity by partitioning the design
Disadvantages of Buses

- It creates a communication bottleneck
  - The bandwidth of that bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
  - The length of the bus
  - The number of devices on the bus
  - The need to support a range of devices with:
    - Widely varying latencies
    - Widely varying data transfer rates

Types of buses

- Processor-Memory Bus (design specific)
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
    - Connects directly to the processor
    - Optimized for cache block transfers
- I/O Bus (industry standard)
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
- Backplane Bus (standard or proprietary)
  - Backplane: an interconnection structure within the chassis
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one bus for all components

One Bus (Backplane) Architecture

- A single bus (the backplane bus) is used for:
  - Processor to memory communication
  - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC - AT

Two Bus System

- I/O buses tap into the processor-memory bus via bus adaptors:
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices
Three Bus System

- A small number of backplane buses tap into the processor-memory bus
  - Processor-memory bus is used for processor memory traffic
  - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

Bus Transaction

- Bus-masters initiate bus transactions
  - E.g. processor issues memory requests
- Slave devices can only respond
  - Never initiate a bus transaction
  - E.g. memory

Synchronous vs. Asynchronous

- **Synchronous Bus**:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast
- **Asynchronous Bus**:
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol
Simple Synchronous Protocol

- Even memory buses are more complex than this
  - memory (slave) may take time to respond
  - need to control data rate

Typical Synchronous Protocol

- Slave indicates when it is prepared for data xfer
- Actual transfer goes at bus rate

Asynchronous Bus Protocol

- Proc asserts READREQ, <address>
- Mem asserts ACK after reading <address>
- Proc deasserts READREQ, <address> and waits
- Mem asserts DATA-RDY, <data>
- Proc asserts ACK after reading <data>
- Mem deasserts DATA-RDY, <data>
- Proc deasserts ACK

FSM control for Asynchronous Bus

- Proc asserts READREQ, <address>
- Mem asserts ACK after reading <address>
- Proc deasserts READREQ, <address> and waits
- Mem asserts DATA-RDY, <data>
- Proc asserts ACK after reading <data>
- Mem deasserts DATA-RDY, <data>
- Proc deasserts ACK
Increasing the Bus Bandwidth

- **Separate versus multiplexed address and data lines:**
  - Address and data can be transmitted in one bus cycle if separate address and data lines are available.
  - Cost: (a) more bus lines, (b) increased complexity.

- **Data bus width:**
  - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles.
  - Example: SPARCstation 20's memory bus is 128 bit wide.
  - Cost: more bus lines.

- **Block transfers:**
  - Allow the bus to transfer multiple words in back-to-back bus cycles.
  - Only one address needs to be sent at the beginning.
  - The bus is not released until the last word is transferred.
  - Cost: (a) increased complexity, (b) decreased response time for request.

Multiple Bus Masters-Arbitration

- **Bus arbitration scheme:**
  - A bus master wanting to use the bus asserts the bus request.
  - A bus master cannot use the bus until its request is granted.
  - A bus master must signal to the arbiter after finish using the bus.

- **Bus arbitration schemes usually try to balance two factors:**
  - Bus priority: the highest priority device should be serviced first.
  - Fairness: Even the lowest priority device should never be completely locked out from the bus.

- **Bus arbitration schemes can be divided into four broad classes:**
  - Daisy chain arbitration: single device with all request lines.
  - Centralized, parallel arbitration: see next slide.
  - Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
  - Distributed arbitration by collision detection: Ethernet uses this.

---

**Daisy Chain Arbitration**

- **Advantage:** simple.
- **Disadvantages:**
  - *Cannot assure fairness:*
    - A low-priority device may be locked out indefinitely.
  - The use of the daisy chain grant signal also limits the bus speed.

**Centralized, Parallel Arbitration**

- **Used in essentially all processor-memory busses and in high-speed I/O busses.**
Distributed Arbitration

- Two flavors
  - Self selection
    - All devices interested in bus assert their requests
    - The contenders for the bus are visible to all devices
    - Each device independently arrives at the same conclusion (priority/round-robin etc)
  - Collision detection with backoff
    - Optimistic attempt to use the bus
    - Can detect collision
    - All colliding parties back-off for random (and increasing) intervals

Increasing Transaction Rate on Multimaster Bus

- Overlapped arbitration
  - perform arbitration for next transaction during current transaction
- Bus parking
  - master can hold onto bus and performs multiple transactions as long as no other master makes request
- Overlapped address / data phases
  - requires one of the above techniques
  - Pipelined bus
- Split-phase (or packet switched) bus
  - completely separate address and data phases
  - arbitrate separately for each
  - address phase yield a tag which is matched with data phase
- "All of the above" in most modern mem buses

Real Stuff: PCI

- Dual Independent Bus (DIB) Architecture between processor/memory/L2 cache
- Peripheral Component Interconnect
  - 33 (66) MHz, synchronous, 32 (64) bit data, multiple masters
- Claim: PCI too slow for high-end graphics.
  - AGP taps directly into FSB

Outline

- How does the CPU initiate I/O?
- How does I/O device notify CPU when I/O op is complete?
- Who (on the CPU) performs I/O ops?
I/O Interfacing

- I/O operation needs to be initiated
  - Special opcodes
  - Memory-mapped I/O
- I/O completion must be known
  - Polling
  - Interrupt-driven

I/O commands

- Memory-mapped I/O
  - Special addresses not for memory ($r9$ contains special address for IOC)
  - Commands written (sw) as data ($r4$)
- I/O commands
  - Special opcodes
  - Send over I/O Bus

I/O Device Notifying the OS

- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error
- This can be accomplished in two different ways:
  - Polling:
    - The I/O device put information in a status register
    - The OS periodically check the status register
  - I/O Interrupt:
    - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.

Polling: Programmed I/O

- Advantage:
  - Simple: the processor is totally in control and does all the work
- Disadvantage:
  - Polling overhead can consume a lot of CPU time

Busy wait loop not an efficient way to use the CPU unless the device is very fast!

but checks for I/O completion can be dispersed among computation intensive code
Interrupt Driven Data Transfer

- Advantage:
  - User program progress is only halted during actual transfer
- Disadvantage, special hardware is needed to:
  - Cause an interrupt (I/O device)
  - Detect an interrupt (processor)
  - Save the proper states to resume after the interrupt (processor)

Direct Memory Access (DMA)

- Direct Memory Access (DMA):
  - External to the CPU
  - Act as a master on the bus
  - Transfer blocks of data to/from memory without CPU intervention

DMA interfacing

- DMA: virtual addresses or physical addresses?
- Challenge: crossing page boundaries
  - Virtual addresses
    - Translation provided by OS
  - Physical addresses
    - One page per transfer
    - OS chains the physical addresses
- No page faults in between
  - "Pin" pages
Delegating I/O Responsibility

- Device transfers are controlled by the IOP directly.
- IOP steals memory cycles.

The Coherence Problem

- Where is the cache with respect to I/O device (and/or I/O processor)?
  - In front of cache: Slows down CPU
  - Behind cache: Cache coherence

OS Responsibilities

- The operating system acts as the interface between:
  - The I/O hardware and the program that requests I/O
  - Why?
    - Why not let user programs directly read/write to I/O devices?

- Three characteristics of the I/O systems:
  - The I/O system is shared by multiple program using the processor:
  - I/O systems often use interrupts (external generated exceptions) to communicate information about I/O operations:
    - Intermits must be handled by the OS because they cause a transfer to supervisor mode
    - The low-level control of an I/O device is complex:
      - Managing a set of concurrent events
      - The requirements for correct device control are very detailed
Interfacing: Summary

- Multiprogramming
  - Program invokes syscall (i.e., invokes OS)
  - OS checks protection
  - OS runs device drivers
  - Suspends current process and switches process
  - I/O interrupt fielded by OS
  - OS completes I/O and wakes up suspended process (i.e. make it runnable)
  - Run next ready process

Video

- Frame-buffer
  - Memory representation of monitor contents
  - Color of each pixel
    - E.g., 8 bits/color = 24 bits/pixel
    - # of colors = 256 x 256 x 256 = 16,777,216 = 16M colors

"Dumb" Video Adapter

- CPU responsible for all data conversion from internal symbolic data-structures to final pixel representation
  - Rendering
    - Frame rate
  - Video adapter grabs data from frame buffer
    - Refresh rate
  - Bandwidth:
    - 24 bit color, 1280x1024 pixels, 75Hz refresh rate
    - 3 x 1280 x 1024 x 75 = 554 MB/s
  - On memory bus
    - Structural hazard
    - CPU and Video adapter competing for memory bandwidth

Optimization #1

- VRAM (Video RAM)
  - Eliminate structural hazard
  - Dual ported
    - Normal DRAM port
    - Serial port
      - Rather scan -> read in serial (row) order
      - Natural because DRAM read in rows
    - Latch entire row in a shift-register
Optimization #2

- Improve Frame rate
  - Smart video accelerators
  - Graphics co-processor
  - Offload rendering (at least significant fractions of it)

- Fringe benefit
  - Massive reduction in bandwidth
  - CPU not writing frame buffer
  - CPU issues graphics commands through API
    - APIs such as OpenGL, Direct3D etc.

Optimization #3

- How OpenGL/Direct3D is implemented on the Graphics Co-processor:
  - Interface vs. Implementation
    - API specifies interface
    - Software implementation
  - Runs on Graphics co-processor
  - Hardware implementations
    - Support for clipping, rendering, transformations etc

Other Accelerators

- Accelerators
  - Per application
    - OpenGL accelerators for 3D rendering
    - Mpeg decoders for dvd (movie) rendering
Optical Drives

- **CD-ROM vs. DVD-ROM**
  - Best as Read-only media
    - CD-R/RW etc. are NOT read/write media in the traditional sense
  - 700 MB vs. 4 GB (maybe more with multiple sides/layers)
  - Pits/bumps, reading laser
  - Nothing fundamentally different in DVD
    - Evolutionary changes
      - Tighter layout, smaller pits
      - Less redundancy for error message
      - Multiple layers/sides

- **CD-R/RW etc.**
  - CD-R/RW etc. are NOT read/write media in the traditional sense
  - 700 MB vs. 4 GB (maybe more with multiple sides/layers)
  - Pits/bumps, reading laser
  - Nothing fundamentally different in DVD
    - Evolutionary changes
      - Tighter layout, smaller pits
      - Less redundancy for error message
      - Multiple layers/sides

Network

- **WAN**
  - Backbone
- **LAN**
  - Ethernet
  - WLAN (802.11b)

Ethernet

- **One shared medium**
  - Distributed arbitration by collision detection
  - Officially: **CSMA/CD**: carrier-sense multiple access with collision detection
  - Length limits (why?)
    - Shared medium is bottleneck
  - "The rumors of my demise have been greatly exaggerated." -- Mark Twain
- **Evolving**
  - 100M Bit ethernet, Gigabit ethernet
  - Now a switched network standard
    - Not shared bus, point-to-point network

802.11b

- **Also distributed arbitration**
  - **CSMA/CA**: carrier-sense multiple access with collision avoidance
  - To minimize collision:
    - Combination of
      - Handshakes (RTS/CTS)
      - Backoff
  - Shared medium is the 802.11b spectrum (near 2.4GHz)
  - Cannot detect collision
    - Limitation of RF: turn off receiver when transmitting
  - Destination has to send an acknowledgement
    - No Ack -> collision and/or RF interference
Networking I/O Issues

• Protocol Stack
  – Application: sends a HTML file
  – HTML file sent over TCP/IP
  – TCP breaks up html file into IP packets
  – Sends IP packets (and ensures in-order delivery, retransmitting if necessary)
  – IP packets may further be broken up into Ethernet/802.11b frames
• Checksums/ error handling at several layer

Networking I/O Issues

• I/O Issues
  • Partitioning this stack
    – What gets done on the CPU?
    – What gets done on other hardware?
• TCP offload engines (TOE)
  – "smart NIC" handles more of TCP stack functionality
  – Relevant at GigE speeds
• Other functionality may also be offloaded
  – Firewall/VPN/filtering

Summary

• Disks
• Buses
  – Performance vs. Simplicity tradeoff
  – Arbitration
  – Timing
• Interfacing
  – OS vs. user mode, virtualized interface
• Video, Networks and Optical Drives